18.453 Lecture 3

Lecture plan

1. Min-weight perfect matching

2. Linear/integer program formulation

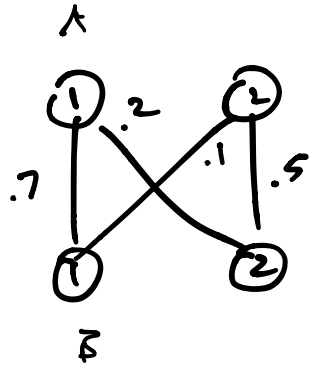
3. Primal-Dual algorithm.

Minimum Weight Perfect Matching

Consider bipartite graph with $|A| = |B| = \frac{n}{2}$

edge ij costs $c_{ij} \in \mathbb{R}$.

E.g.



$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} .7 & .2 \\ .1 & .5 \end{bmatrix}$$

Goal: find matching M in G of least cost

$$c(M) := \sum_{ij \in M} c_{ij}.$$

by allowing $c_{ij} = \infty$, can assume G is complete bipartite graph.

Application: n machines, n tasks, costs c_{ij} for machine i to do task j.

Today: Hungarian algorithm

- uses linear programming
- is strongly polynomial time:
  # steps independent of sizes of c_{ij}; polynomial in n.

Linear/integer programs

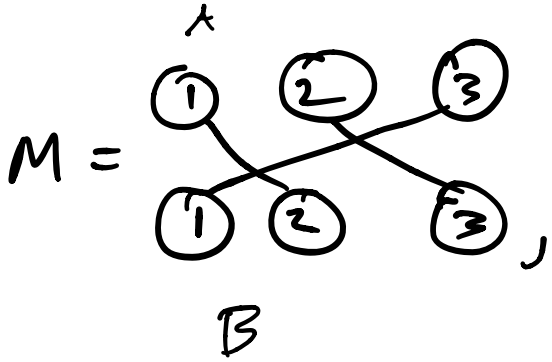
- First, express problem as integer program.
- Associate vector with matching.

  incidence vector of matching M is vector x s.t.

$$x_{ij} = \begin{cases} 1 & \text{if } ij \in M \\ 0 & \text{else.} \end{cases}$$

(confusingly, also a matrix)

E.g.



$$M = \{ \quad , \quad \}$$

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

note: X permutation matrix.

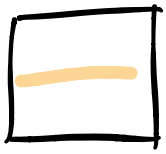
Integer program: (IP)

min-weight perfect matching has cost

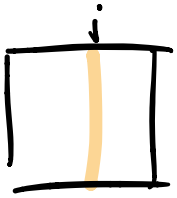
$$\min \sum c_{ij} x_{ij}$$

Subject to

$$\sum_j x_{ij} = 1 \quad \forall i \in A$$



$$\sum_i x_{ij} = 1 \quad \forall j \in B$$



$$x_{ij} \geq 0 \quad \forall i \in A, j \in B$$

$$x_{ij} \in \mathbb{Z} \quad \forall i \in A, j \in B$$

not a linear program b/c

Any solution to IP is valid matching & vice versa.

Linear program (LP)

Get linear program (P) by dropping integrality constraint.

$$
\begin{aligned}
&\min \sum c_{ij} x_{ij} && \text{"objective"}\\
&\text{subject to } \sum_j x_{ij} = 1 && \forall i \in A \\
&\qquad\qquad \sum_i x_{ij} = 1 && \forall j \in B \\
&\qquad\qquad x_{ij} \geq 0 && \forall i \in A, j \in B
\end{aligned}
\qquad (P)
$$

"constraints"

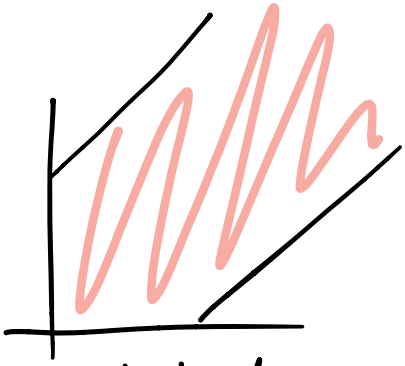
Called the linear programming relaxation of the integer program.

Say x **feasible** if satisfies constraints.

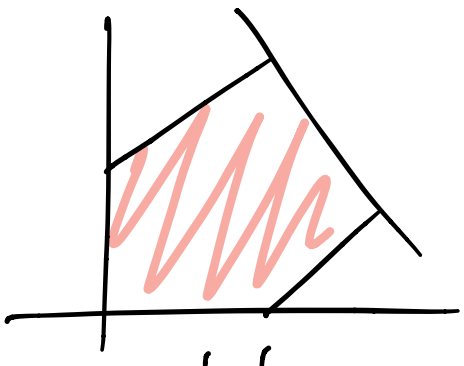
In contrast to IP: not all feasible x are matchings!

x_{ij} can be fractional.

Set of feasible solutions is a **polytope** (bounded polyhedron).



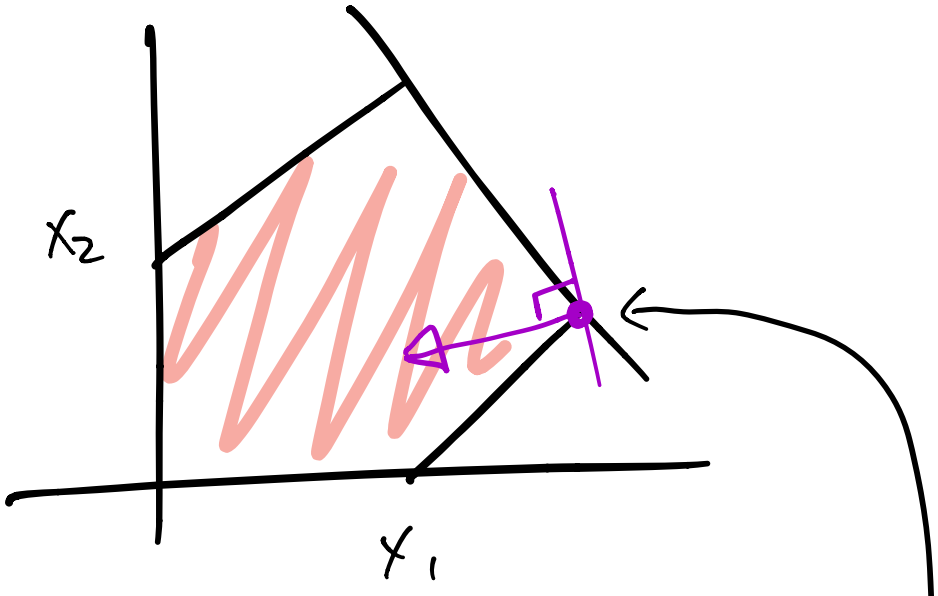
polyhedron



polytope

optimum of a linear function will occur at an extreme point (corner).

E.g. if $c =$ 

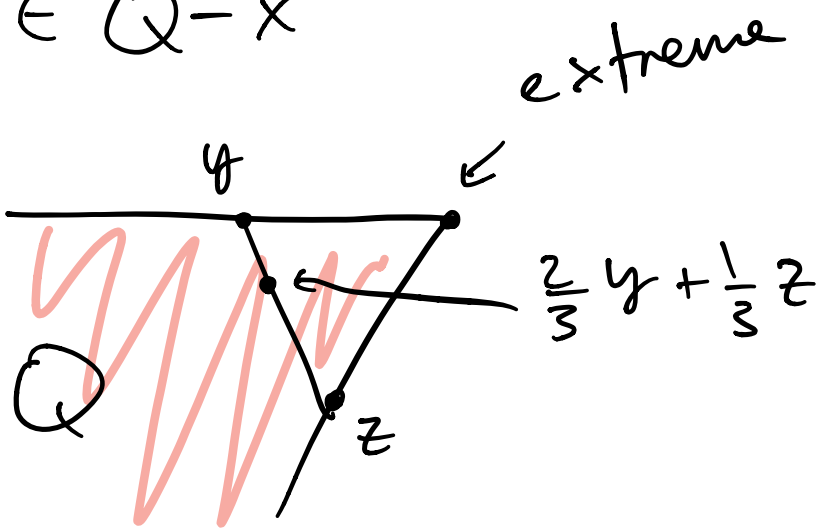


minimizes $c \cdot x$ over polytope.

Extreme point x of set Q is point that can't be written as

$$\lambda y + (1-\lambda) z \quad , \lambda \in (0,1)$$

for $y, z \in Q - x$



(more on this when we get to polyhedral combinatorics).

In general, extreme points need not be integral (even if constraints all have

coefficients in $\{0,1\}$.)

No surprise: L.P. solvable in polynomial time, I.P. NP-hard.

Say Z_{IP} = value of some IP

Z_{LP} = value of its relaxation;

In general

$$Z_{IP} \neq Z_{LP}.$$

But! IP is more constrained, so

$$Z_{IP} \geq Z_{LP}.$$

for minimization problems.

Moreover: if x is optimum for LP, and x integral, then x opt for IP!

Exercises: 1. prove this ↑

2. find example where $z_{IP} \neq z_{LP}$.

For perfect matching, we are lucky! Constraints special.

Consider the polytope P

cutout by constraints of (P).

$$P = \{ x \text{ s.t.}$$

$$\sum_j x_{ij} = 1 \quad \forall i \in A$$

$$\sum_i x_{ij} = 1 \quad \forall j \in B$$

$$x_{ij} \geq 0 \quad \forall i \in A, j \in B \}$$

Theorem: every extreme point of P is integral.

(in particular, is a 0-1 vector and hence is the incidence matrix of P.M.).

We give 2 proofs:

1. algorithmic (today)
2. algebraic (later); uses total unimodularity

First: duality for LP's.

(informal version).

LP duality

Dual of (P): family of obstructions for (P) to have small value.

Recall:

(P)

$$\min \sum c_{ij} x_{ij}$$

Subject to

$$\sum_j x_{ij} = 1 \quad \forall i \in A$$

$$\sum_i x_{ij} = 1 \quad \forall j \in B$$

$$x_{ij} \geq 0 \quad \forall i \in A, j \in B$$

obstruction: values

$$u_i \quad i \in A,$$

$$v_j \quad j \in B$$

s.t. $u_i + v_j \leq c_{ij} \quad \forall i \in A, \forall j \in B.$

Then: for any matching M,

$$\sum_{ij \in M} c_{ij} \geq \sum_{ij \in M} u_i + v_j = \boxed{\sum_{i \in A} u_i + \sum_{j \in B} v_j}.$$

↑ this value is our obstruction.

want to <u>maximize</u> this value; doing this gives us the dual (D) of (P).

$$\max \sum_{i \in A} u_i + \sum_{j \in B} v_j$$

$$(D) \quad u_i + v_j \leq c_{ij} \quad \forall i \in A, \ \forall j \in B.$$

In fact, $\sum_{i \in A} u_i + \sum_{j \in B} v_j$ is not just a lower bound on $c(M)$, but on (P).

Indeed, we can calculate:

$$\sum_{ij} c_{ij} x_{ij} \geq \sum_{ij} (u_i + v_j) x_{ij}$$

$$= \sum_{ij} u_i x_{ij} + \sum_{ij} v_j x_{ij}$$

$$= \sum_{i \in A} u_i \left(\sum_{j \in B} x_{ij} \right)$$

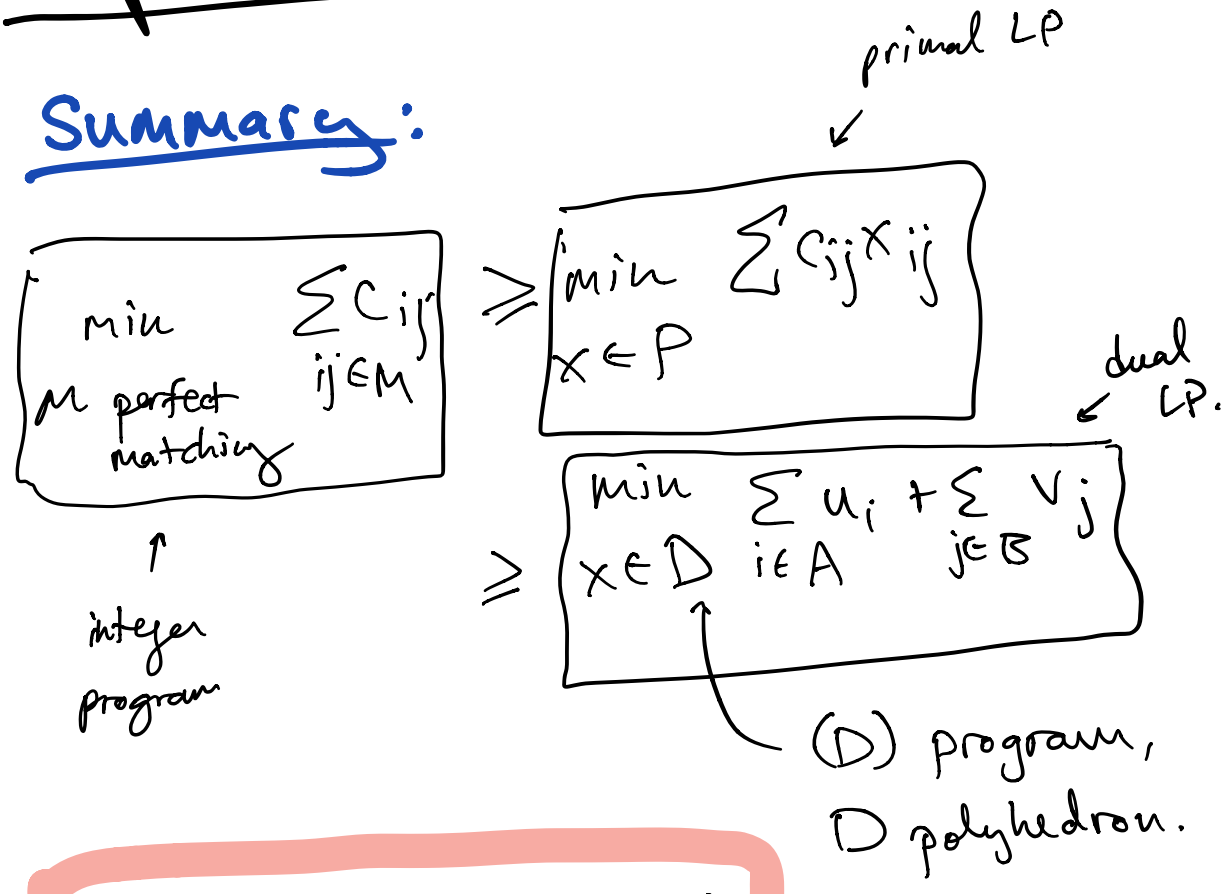
$$+ \sum_{j \in B} v_j \left(\sum_{i \in A} x_{ij} \right)$$

$$= \sum_{i \in A} u_i + \sum_{j \in B} v_j .$$

Construction $(P) \rightsquigarrow (D)$ is example of more

general "recipe" for taking dual of LP's.

Summary:



When equality ??

M must only have edges (i,j) s.t.

$$c_{ij} = u_i + v_j .$$

"complementary slackness" in LP lingo.

- Let $w_{ij} := c_{ij} - u_i - v_j$.
- Are matchings on $\{(i,j) : w_{ij} = 0\}$, but no guarantee they are perfect.
- primal-dual alg uses such (non-perfect matchings) to update dual solution u_i, v_j.

Primal-Dual

Outline:

1. start w/ any dual feasible solution, e.g. $u = 0$, $v_j = \min_i c_{ij}$.

Repeat the following until done:

2. In any iteration, alg. has dual feas. soln. (u, v) or (u, v, w).

3. Want a matching on

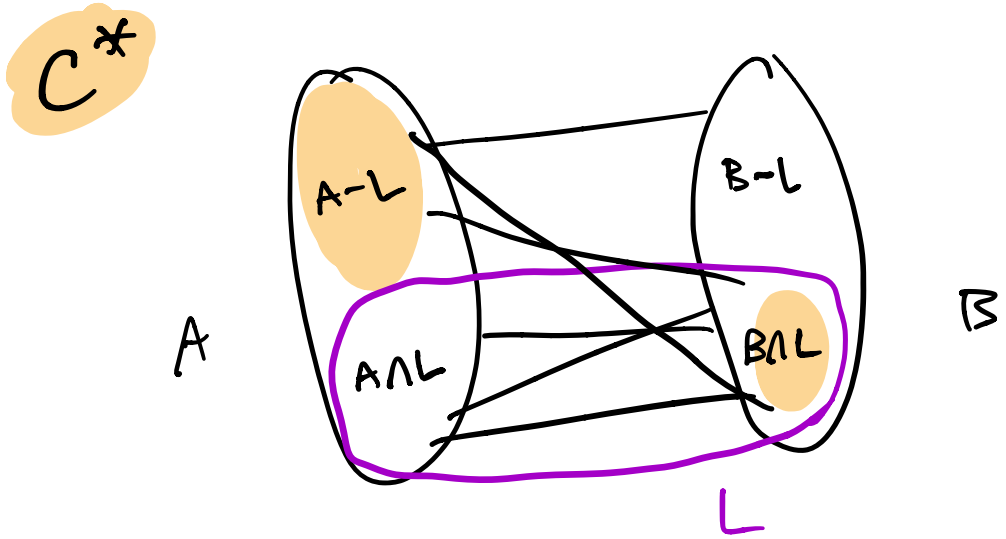
$$E = \{(i,j) : w_{ij} = 0\}.$$

Use cardinality matching alg. to output largest M.

- If M perfect, is optimal by complementary slackness.
- If not, use the vertex cover output by alg. to find new dual feasible soln w/ larger value.

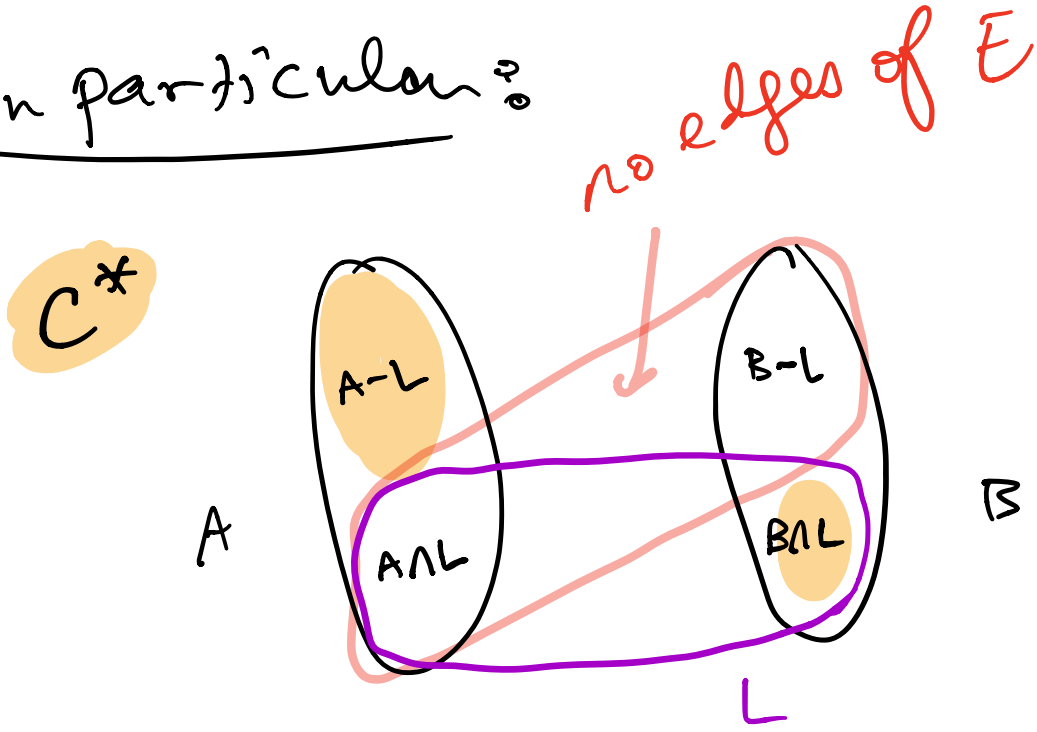
Details of step 3:

- Suppose M not perfect.
- Recall set L output by the aug. paths algorithm.



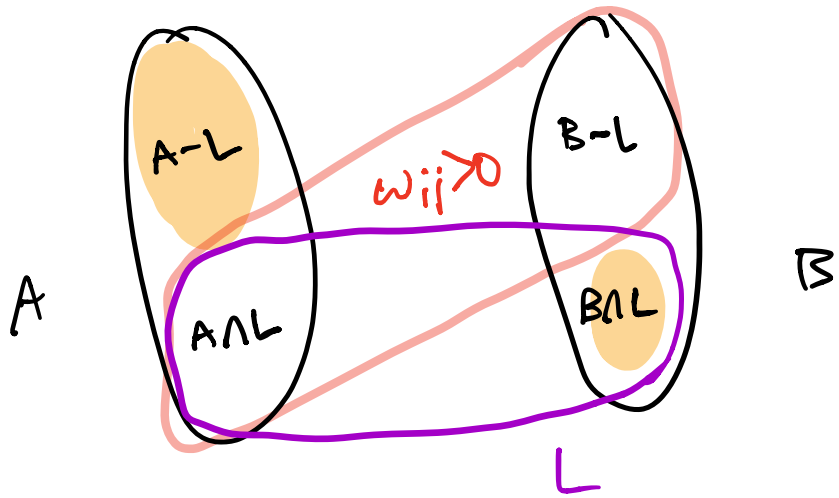
$C^* = (A - L) \cup (B \cap L)$ is optimal vertex cover of E.

In particular:



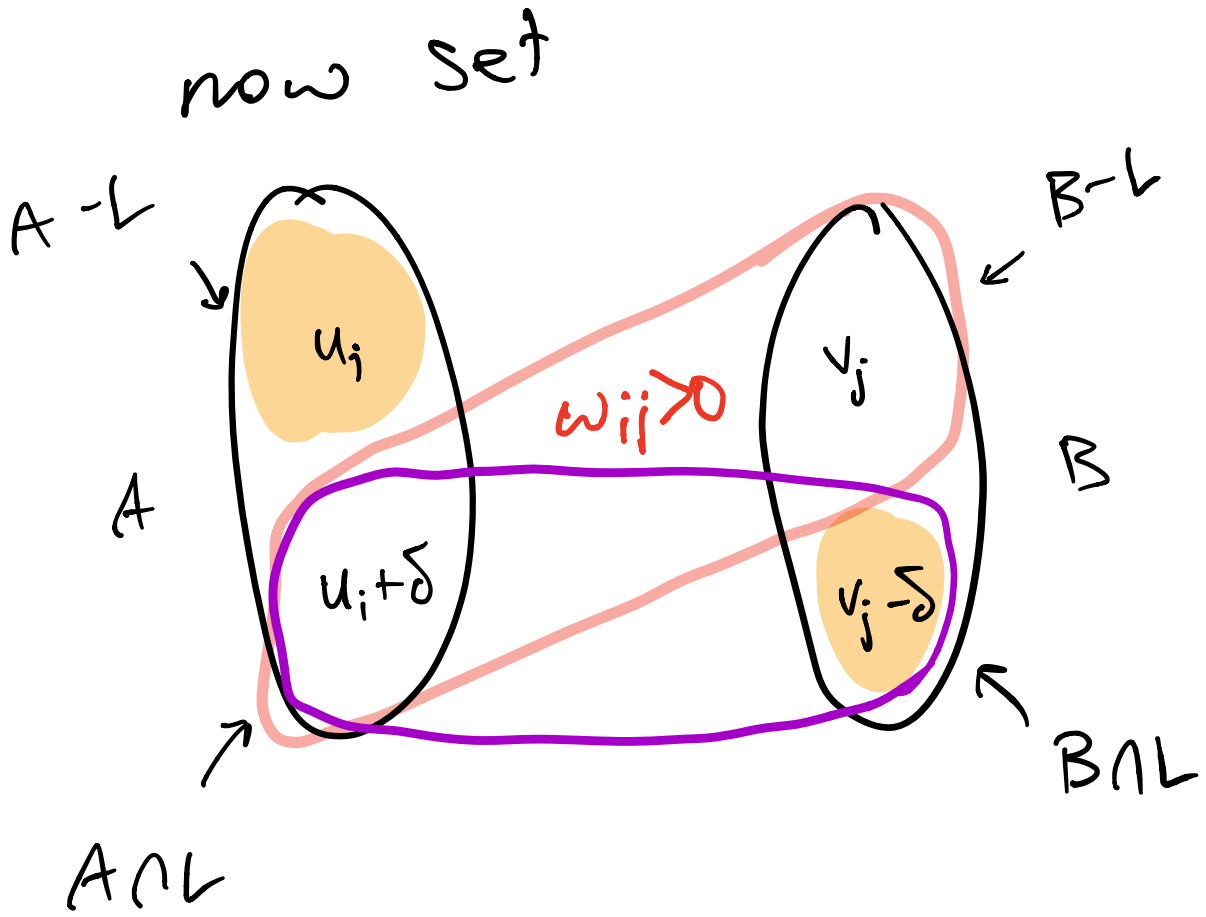
Equivalently: $w_{ij} > 0$ for

$i \in A \cap L$, $j \in B - L$.



Updating u, v:

$$\delta = \min_{\substack{i \in (A \cap L) \\ j \in (B - L)}} w_{ij}, \qquad (\delta > 0)$$



formally:

$$u_i = \begin{cases} u_i & i \in A - L \\ u_i + \delta & i \in A \cap L \end{cases}$$

$$v_j = \begin{cases} v_j & i \in B - L \\ v_j - \delta & j \in B - L. \end{cases}$$

New solution is **feasible!**

New value?

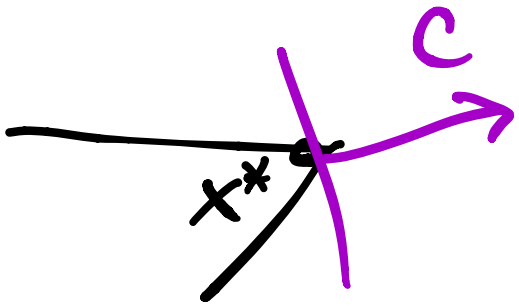
$$\text{New} - \text{OLD} = \delta\left(|A \cap L| - |B \cap L|\right).$$

$$= \delta\Big(\underbrace{|A \cap L| - |A - L|}_{|A|} + \underbrace{|A - L| - |B \cap L|}_{|C^*|}\Big)$$

$$= \delta\left(\frac{n}{2} - |C^*|\right) \geq \delta.$$

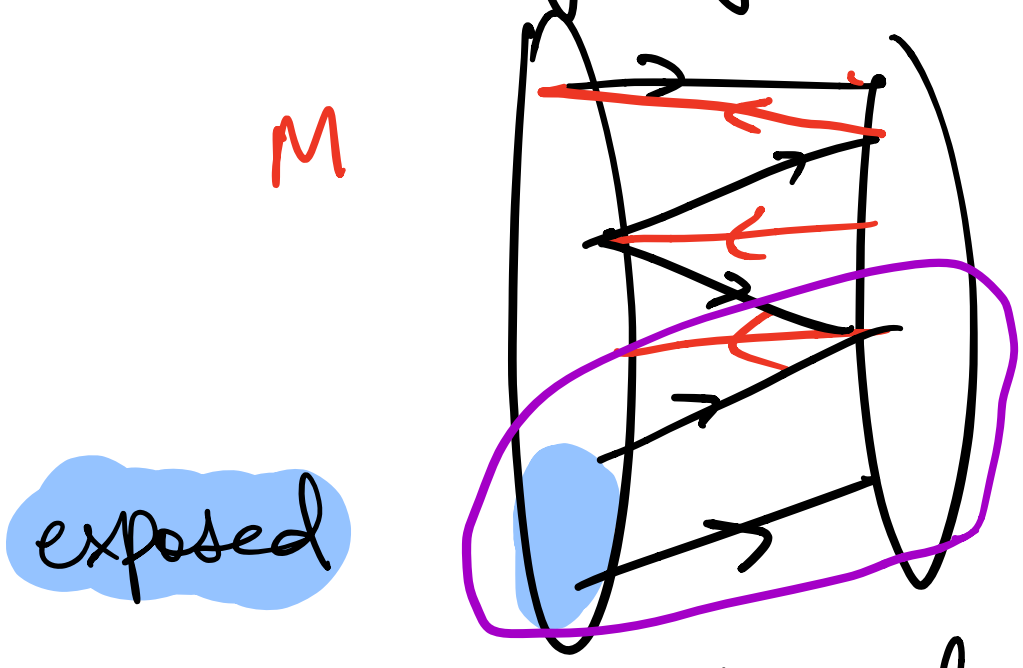
Thus, dual value increases! Repeat until termination - then M is perfect; done.

Proves theorem 16: for any extreme pt x^*, can choose c to make x^* unique optimum.

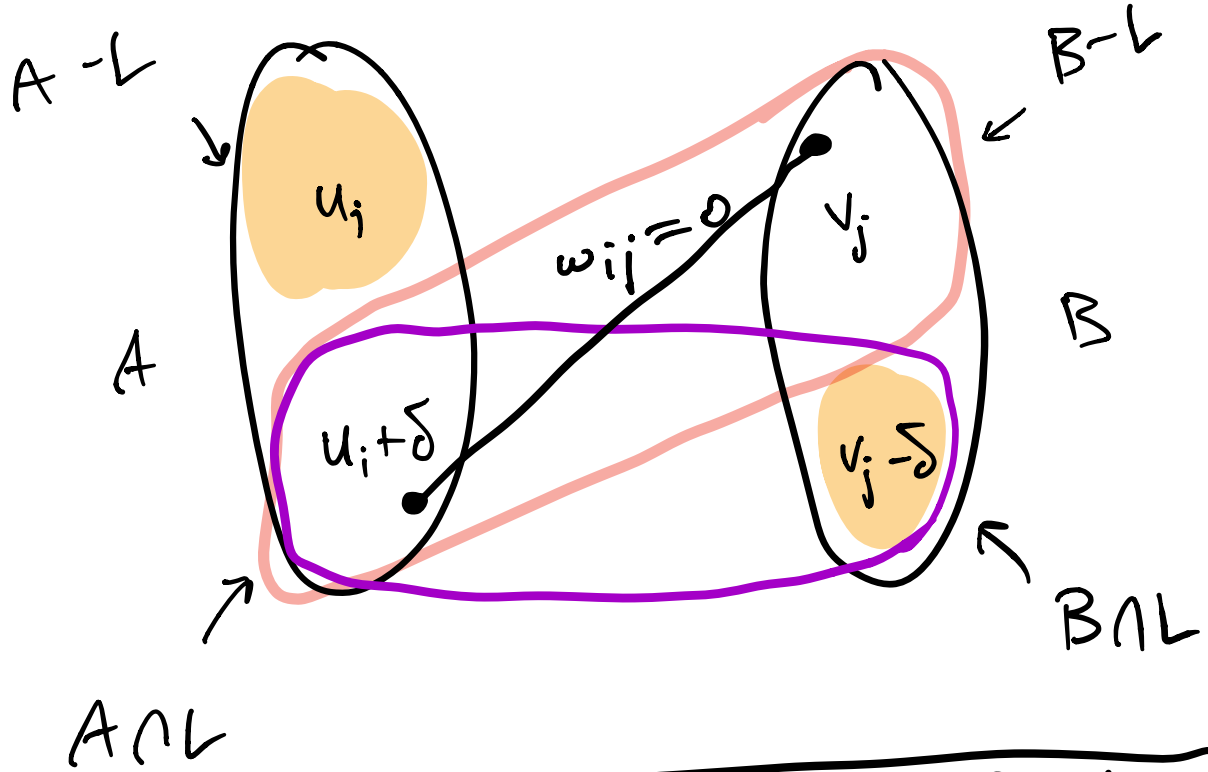


Termination? How do we know it terminates?

Recall def of L.



everything reachable from exposed in A.



for some $i \in A \cap L$, $j \in B - L$, $w_{ij} = 0$ by our choice of δ.

New vertex j reachable.

thus, in $\leq \frac{n}{2}$ iterations either

- an exposed vertex in T_S is reached and $|M|$ can increase by 1.

- all verts of R reachable, none exposed (M perfect).

$\leq \frac{n}{2}$ iterations per M,

$\leq \frac{n}{2}$ new M's $= O(n^2)$ iterations.

Overall running time

$O(n^4)$

b/c takes $O(n^2)$ time to compute L. $\square$

Exercise: By tracking more carefully how L changes, show $O(n^3)$.

Remark: strongly polynomial time: poly in n, assuming arith. operations free.